EXHIBIT 25

Non-exhaustive search methods and their use in the minimization of Reed–Muller canonical expansions

G. I. ROBERTSON†, J. F. MILLER† and P. THOMSON†

A number of non-exhaustive search algorithms are presented. The methods are a 'classical' genetic algorithm, a tabu search, an evolutionary strategy and stochastically repeated nearest and steepest-ascent hill-climbing algorithms. They are then used to determine optimum and good polarities for Reed–Muller canonical expansions of Boolean functions, and comparisons are drawn between the relative effectiveness of each method. Tabu search and nearest-ascent hill-climbers are found to be particularly appropriate for these problems.

1. Introduction

This paper is, in part, a follow-up to a previous paper (Miller *et al.* 1994), which examined the use of a genetic algorithm (GA) in finding the best polarity for Reed–Muller canonical (RMC) expansions, and, in part, an investigation into the application of alternative, modern optimization methods to these problems. The problem with RMC expansions is in choosing that expansion which minimizes the number of required logic gates to implement the input function. We chose to test these optimization techniques on the Reed–Muller fixed polarity problem, as the nature of the search space is both manageable and well understood and it is a problem with which we are well acquainted (Almaini *et al.* 1991, McKenzie *et al.* 1993, Miller and Thomson 1994).

Boolean logic functions may be expressed in terms of Boolean algebra (inclusive-OR and AND gates) or as modulo-2 sums-of-products (exclusive-OR and AND gates) (Green 1986, 1987, Almaini 1989) commonly known as Reed–Muller algebra. Implementation of logic functions in Reed–Muller have two advantages. First, Reed–Muller circuits are readily testable (Reddy 1972). Secondly, it is often the case that logic functions that do not minimize well in the Boolean domain can be reduced substantially if implemented in Reed–Muller logic. The most general representation of a Reed–Muller logic function is an exclusive-OR sum-of-products. In this representation, variables may be complemented and uncomplemented in the same expansion without restriction. This allows an enormous number of equivalent representations. In fixed polarity or canonical (RMC) expansions the logic variables must be either complemented or uncomplemented throughout the entire expression. This reduces the number of possible expansions to 2^n. In this paper we deal only with the latter.

Techniques for conversion of Boolean expressions in minterm form are well established (Mukhopadhyay and Schmitz 1970, Besslich 1983, Tran 1987, Almaini *et al.* 1991). A number of exhaustive search techniques for global best polarity have been devised (Harking 1990, Almaini *et al.* 1991, Lui and Muzio 1991, Sarabi and Perkowski 1992, Miller and Thomson 1994). The most efficient of these have time

Received 1 July 1995; accepted 12 July 1995.

† Department of Electrical, Electronic and Computer Engineering, Napier University, 219 Colinton Road, Edinburgh EH14 1DJ, U.K. e-mail: j.miller@napier.ac.uk.

0020–7217/96 $12·00

complexities of $O[3^n]$ and are presently impractical for more than $n=14$ variables. In response to this, a number of deterministic algorithms have been developed which find good sub-optimum polarities in a much reduced time. McKenzie *et al.* (1993) compared the performances of a number of such algorithms. They found an original algorithm (referred to in their paper as 'Gains') to be the most effective on both commonly used benchmark examples (see, for example, Sarabi and Perkowski 1992) and randomly generated problems of up to 12 variables, for which the basis of comparison was grading against exhaustive search results. As we reported in our last paper in this area (Miller *et al.* 1994) the GA significantly outperformed these other methods.

During the development of our original GA for this class of problem an *ad hoc* optimization of the population size and mutation rate parameters was carried out. This led to a population size far smaller and a mutation rate far higher than those generally accepted to be normal by the genetic algorithm community (e.g. Goldberg 1989). Furthermore, the preliminary application of a metric developed by the authors, based upon the epistasis variance method (Davidor 1991) to randomly generated Boolean functions, suggested that the functions possess a low degree of nonlinearity. Davidor asserts that GAs are most appropriate for functions with a medium degree of nonlinearity and that low epistasis problems are more efficiently tackled using hill-climbers. The low degree of epistasis may indeed be related to the fact that RMC polynomials have a great deal of mathematical structure so that it is even possible for small variable problems (three variables) to obtain exact formulae for optimum polarities (Miller and Thomson 1994). As a preliminary to further work on optimizing the GA it was decided to carry out a systematic comparison with some other non-exhaustive search methods.

2. The search algorithms

The fixed polarity problem lends itself to a straightforward representation suitable for all the search algorithms. A polarity is encoded as a string of bits (genes) of length n, where n is the number of functional input variables. The meaning attached to a chromosome $b_{n-1}, b_{n-2}, \ldots, b_0$ representing a particular polarity is as follows: if $b_i=1$ or 0 then variable x_i will be complemented or not complemented, respectively, throughout the expansion associated with the particular polarity.

A chromosome is evaluated for a particular Boolean function by an algorithm known as the fitness function. The fitness function employs the tabular method (Almaini *et al.* 1991) to determine gate count and is the same as that used previously (Miller *et al.* 1994). The fitness assigned to a particular chromosome associated with a polarity is given by 2^n minus the number of XOR operations required to express the Boolean function in modulo-2 algebra using this polarity. The search algorithms seek to find chromosomes (polarities) with high fitnesses for the particular Boolean input function being examined. For an n variable problem, the search space (the set of all possible polarities) has 2^n elements and the time complexity for fitness function evaluation is $O[2^n]$.

All the search algorithms used are blind: they use no *a priori* knowledge of the optimization landscape and are guided solely by information provided by the fitness function. They are thus generic and, by choice of an appropriate fitness function and chromosome representation, are applicable to a wide range of problems. In contrast to previous, deterministic approaches to the RMC minimization problem, they all

contain at least some stochastic element. Thus, the quality of the solution obtained is (on average) dependent on the length of time for which an algorithm runs.

In the algorithm descriptions shown below, the termination conditions have been omitted. This was done partly for the sake of simplicity and partly because we investigated two different sets of termination criteria. These are discussed later.

2.1. *Neighbourhood methods*

The steepest-ascent hill-climber (SAHC), nearest-ascent hill-climber (NAHC) and Tabu Search (TS) are all members of the neighbourhood search (NS) family of algorithms. When the aim of a search is to minimize a cost function, rather than maximize a fitness function, the SAHC and NAHC are known as steepest-descent and nearest-descent methods, respectively. A recent introductory treatment of neighbourhood search methods can be found in Reeves (1993).

If the set of all possible chromosomes is denoted by X and a particular chromosome by $x \in X$ then an associated set of neighbours, $N(x) \subset X$, is called the neighbourhood of x. Each chromosome, $x' \in N(x)$ can be reached directly from x by an operation known as a *move* which, in this context, is an inversion of a single gene. For the SAHC and NAHC, $N(x)$ is the set of all chromosomes at a Hamming distance of one from x.

Tabu search (TS) is a recent addition to the NS family. The central theme underlying this technique is the idea of incorporating memory into the search. A selective history, or tabu list, T, of the chromosomes encountered during the search is maintained. In this scheme $N(x)$ is replaced by a more restricted neighbourhood, $N_R(x) = N(x) - T$ where the minus sign denotes the set-theoretic difference operation. A review of tabu search can be found in Glover (1989, 1990). Glover suggested that the general aim of the tabu list is to avoid returning to a previous solution by preventing reverse moves.

It is interesting to note that the Gains algorithm of McKenzie *et al.* is equivalent to a single iteration of a steepest ascent hill-climb starting at polarity 0. The SAHC is sometimes referred to as a 'greedy' algorithm as it always moves to the very best solution in the neighbourhood. We present SAHC, NAHC and TS algorithms next.

Algorithm 1. Steepest-ascent hill-climber algorithm (SAHC)

Step 1. Choose a chromosome at random. Call this *current-hilltop*.

Step 2. Complement (mutate) each bit (gene) in the chromosome and evaluate the fitnesses of the resulting n chromosomes.

Step 3. If any of the resulting chromosomes have a fitness better than *current-hilltop* set *current-hilltop* to that with the highest fitness and go to Step 2.

Step 4. Else if all resulting chromosomes are no fitter than *current-hilltop* (a local optimum has been reached) go to Step 1.

Algorithm 2. Nearest-ascent hill-climber algorithm (NAHC)

Step 1. Choose a chromosome at random. Call this *current-hilltop*.

Step 2. Move along the *current-hilltop* chromosome mutating single bits and evaluating the fitnesses of the resulting chromosomes.

Step 3. As soon as a chromosome fitter than *current-hilltop* is found set *current-hilltop* to this chromosome and go to Step 2, continuing mutation at the bit position after that which gave the fitness improvement.

Step 4. If no chromosome fitter than *current-hilltop* was found (local optimum found) go to Step 1.

Algorithm 3. Tabu search algorithm (TS)

Step 1. Initialize the tabu list, T, to be empty.

Step 2. Randomly choose a chromosome and call this *current-hilltop.*

Step 3. Add *current-hilltop* to the tabu list.

Step 4. Mutate each bit of *current-hilltop* that results in a chromosome that is not on the tabu list and evaluate the fitnesses of these chromosomes. If no chromosomes can be produced (because all chromosomes at a Hamming distance of 1 from *current-hilltop* are on the tabu list) reduce the length of the tabu list from the oldest member towards the most recent until a chromosome results.

Step 5. Set *current-hilltop* to the fittest of the resulting chromosomes (even if this means a decrease in fitness) and go to Step 3.

Glover mentioned that to ensure finiteness, the size of T should be allowed to grow with the number of moves made. However, he also suggested that move reversals may be avoided by making the tabu list consist of solutions rather than explicit moves. This is the procedure we adopted. Our implementation examined all the chromosomes in a neighbourhood that were not on the tabu list before making a move, and thus proceeded aggressively, as is typical in tabu search (Glover 1989).

Several interesting points arise out of this implementation of tabu search and the way that the algorithm behaved on the functions upon which it was tested. We found no occasion during the many times (>2000) it was executed for which all members of the neighbourhood set were on the tabu list, i.e. at no point was $N_R(x)$ equal to the empty set. It can be shown that it is possible to create fitness landscapes which allow such a situation to arise so that the full instructions in Step 4 of Algorithm 3 would have to be applied. It would be interesting to know whether the phenomenon $N_R(x) \neq \varnothing$ is due to: (i) the particular input functions tested; (ii) a matter of chance due to the starting points chosen; or (iii) a general feature of RMC expansions of all logic functions.

The tabu search is the least stochastic technique of those tested: the only random element is the choice of starting chromosome. Thus, the danger of cycling arises: i.e. part of the previous search path may be repeatedly followed. As we allowed the length of our tabu list to grow indefinitely, and as it never became any shorter, since $N_R(x) \neq \varnothing$, cycling could not have occurred during any of our tests. Glover does not unequivocally recommend using an infinitely long tabu list, as doing so may damage performance by causing a greater number of moves to be necessary. However, this reservation is expressed within the context of problems dealing with highly constrained search spaces, such as scheduling and space planning applications. A finite tabu list may also be required due to memory constraints or significant performance overhead being incurred by looking through the tabu list. A technique for detecting and avoiding cycling when using a finite length tabu list can be found in Battiti and Tecchiolli (1992).

2.2. *The genetic algorithm (GA)*

Based in Holland's pioneering study of adaptation in natural and artificial systems (Holland 1968, 1975) genetic algorithms are a non-deterministic approach to problem-solving based on directed adaptation. Good introductions to genetic algorithms can be found in Goldberg (1989), Davis (1991 b) and Michaelowicz (1992). The GA is now well established and there are countless variants. We make no claims for our particular version other than that it is fairly standard. A generic GA may be described by Algorithm 4.

Algorithm 4. The genetic algorithm (GA)

Step 1. Generate the initial population of chromosomes.

Step 2. Evaluate the fitnesses of the present (parent) population.

Step 3. Choose parent chromosomes with probabilities dependent on their fitness.

Step 4. Breed (using crossover) to produce the child population.

Step 5. Apply mutation operator (random bit complementation) to the children.

Step 6. The child population becomes the new parent population. Go to Step 2.

In Step 1, the initial population was generated at random. In Step 3, the probabilities of breeding were assigned linearly with respect to the fittest and least fit in the present population (a windowed proportionate method) so that $f_r = (f - f_{min})/(f_{max} - f_{min})$, where f is the fitness of a chromosome in the population. The probability of a chromosome being selected to produce a child chromosome was directly proportional to its relative fitness. The evolutionary pressure exerted by the method would be regarded as very high (e.g. Hancock 1994) by most GA practitioners and often led to convergence (population homogeneity), of which more later. In Step 4, the two-point crossover method was used.

The GA was implemented with two sets of parameters. The first set of parameters was those we had determined to be effective during the work in our previous paper, and the second set was closer to those considered to be typical (e.g. Goldberg 1989).

GA1: population size=8, probability of mutation (per gene)=0·06
GA2: population size=30, probability of mutation (per gene)=0·03

The probability of crossover was 0·6 in both cases.

2.3. *Evolutionary strategy (ES)*

Evolutionary strategies are another class of algorithms inspired by Darwin's theory of evolution. They emerged independently in Germany (Rechenberg 1973) at about the same time as genetic algorithms in the U.S.A. Evolutionary strategies are more usually used for non-binary representations, i.e. for optimizing problems with more or less continuously varying parameters. In contrast to the GA, the main force behind an ES is mutation; crossover within a parameter is usually regarded as harmful. Parameters on a chromosome are individually mutated by a degree that often follows a normal probability density function. The standard deviation of this density function may be in some way inversely proportional to the fitness of a particular chromosome or may be encoded onto the chromosome and itself subject to modification by the ES. A survey of the development of the ES may be found in Back *et al.* (1991).

Our particular ES is given in Algorithm 5.

Algorithm 5. An evolutionary strategy algorithm (ES)

Step 1. Randomly generate the initial population of N chromosomes and evaluate their fitnesses.

Step 2. Asexual reproduction. Each chromosome produces a single child by probabilistic mutation. The degree of perturbation used to produce the child depends on the fitness of the chromosome relative to others in the present population so that relatively fit chromosomes on average produce children more similar to themselves than do less fit chromosomes.

Step 3. Evaluate the fitnesses of the child chromosomes.

Step 4. If a child chromosome is fitter than its parent it replaces it in the population.

Step 5. Go to Step 2.

The evolutionary strategy we adopted is an example of a $(\mu + \lambda)$-ES since the new population is selected from the old population and its children, rather than from the child chromosomes alone, as is the case in a (μ,λ)-ES. However, our implementation is not typical in that a fitter child chromosome may only replace its own parent—a form of niching behaviour (e.g. Goldberg 1989) designed to retard take-over of the population by very fit, but sub-optimum, individuals.

The probabilistic mutation was implemented in the following way. The mutation was calculated such that the probability of the kth gene being mutated was proportional to e^{-kf_r}. This meant that the fitter individuals in a population initiated more local searching, closely resembling the behaviour of hill-climbers, whereas less fit individuals initiated more widespread exploration. No optimization of the population size (arbitrarily chosen to be 8, in line with GA1) was attempted, nor did we try to ascertain the most advantageous probability density function. The ES was included in the comparison as an example of an algorithm, that tried to keep searching good parts of the search space whilst attempting to move away from poorer parts.

4. Method and results

Each of the algorithms was run on input logic functions of 11, 12, 13, 14 and 15 variables. One-hundred functions were generated for each number of variables giving 500 functions in all. Each function consisted of a random number of random minterms. Duplication of minterms within a function was avoided. Random number generation was carried out both during function generation and execution of the search algorithms using the NAG mathematical libraries.

Two methods for terminating a search were employed, each giving a different measure of relative search algorithm performance: first, termination occurred when a chromosome with the global best fitness was found (exhaustive search for the global best fitness was possible for all functions up to 14 variables); or secondly, by fixing the number of different chromosomes (points visited in the search space) to be evaluated. For the first termination condition, the number of different chromosomes evaluated was recorded for each input function and each algorithm. The results are shown in Fig. 1. For the second termination condition, 50, 100 and 200 different chromosome evaluations were chosen for searches on each of the input functions. The highest fitnesses found were recorded and their means are shown in the Table. The results for the 15 variable input problems were typical and are shown graphically in Fig. 2.

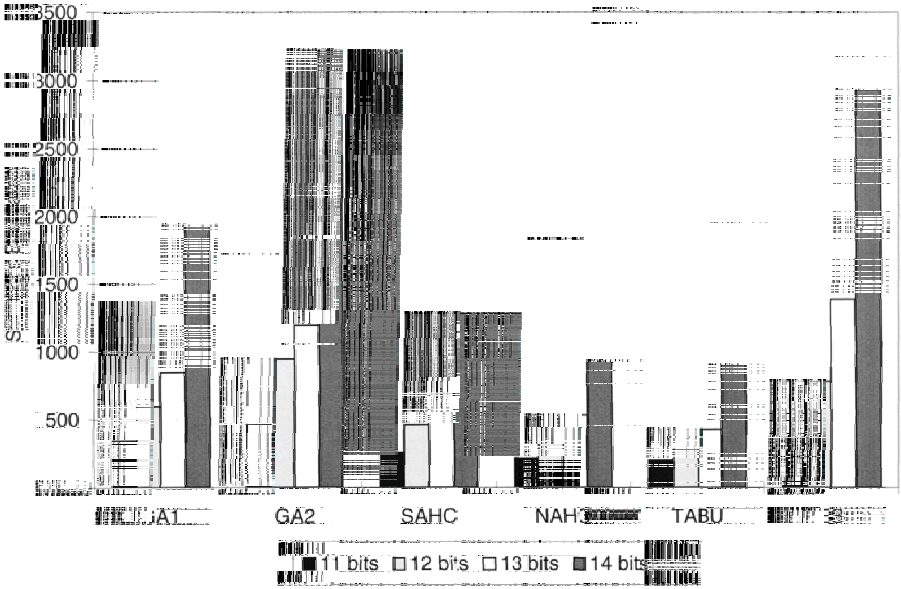

Figure 1. Mean number of chromosomes evaluated to find the best fitness.

No. input variables/ No. evaluations	GA1	GA2	SAHC	NAHC	TABU	ES
11/100	1160	1154	1162	1166	1165	1160
11/200	1165	1164	1169	1172	1171	1169
12/50	2240	2226	2245	2260	2244	2238
12/100	2258	2251	2260	2271	2272	2258
12/200	2272	2266	2278	2280	2277	2276
13/50	4379	4366	4400	4418	4396	4384
13/100	4411	4392	4415	4434	4431	4411
13/200	4435	4429	4439	4450	4452	4441
14/50	8578	8563	8593	8635	8597	8575
14/100	8617	8591	8634	8649	8648	8624
14/200	8643	8643	8662	8677	8672	8653
15/50	16921	16909	16950	17001	16955	16913
15/100	16988	16951	17007	17044	17005	16982
15/200	17020	16996	17051	17084	17072	17049

The mean fitnesses are calculated for 100 random functions of each number of variables. If the number of exclusive-OR operations associated with a particular chromosome (polarity) is denoted by x then the fitness of that chromosome is given by 2^n-x. It is interesting to note that the fitnesses are of the order of 2^{n-1}, thus implying that the *average* least number of XOR operations required to represent a RMC function of n variables is of the order 2^{n-1}.

Mean fitnesses found by algorithms for differing numbers of input variables and evaluations.

The reason for using the number of different chromosomes evaluated as a performance measure is that, first, this enables us to compare, for each algorithm, their relative effectiveness when examining a fixed proportion of the search space. Secondly, fixing the number of different chromosomes to be evaluated is equivalent to predetermining the total CPU time required. For the algorithms and numbers of

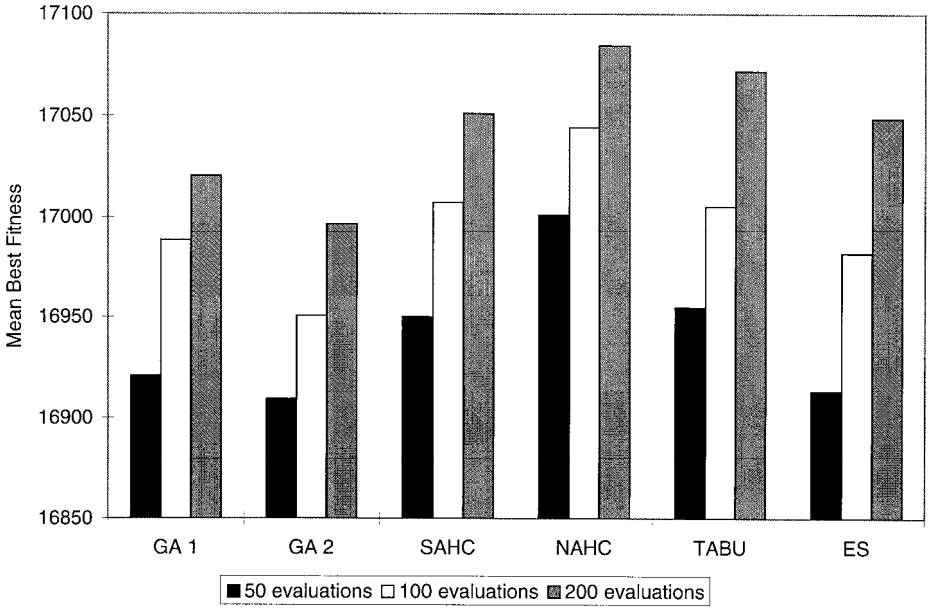

Figure 2. Means of best fitnesses for 15 variable functions.

variables used in these experiments the fitness function execution time was large; the difference between this and the total computation time was found to be negligible. Because of this a look-up table was used whenever the fitness value for a previously evaluated chromosome was required. As the number of variables increases the first termination condition becomes equivalent to the time taken to find the global best solution and the second condition the best fitness achievable in a fixed time. The first termination condition allows a comparison of the algorithms based on the mean proportion of the search space required to find the global best fitness. This is shown in Fig. 3.

5. Discussion of results

On average, the relative performances of the various algorithms were remarkably insensitive to the number of input variables. This was consistent with the trends reported earlier (McKenzie *et al.* 1993) in the comparison of previous algorithms for these problems. We feel that our results may serve as a good guide to the relative average performances of the algorithms for larger numbers of variables.

All the methods used except the GA rely upon the search space having what we might define as the smoothness property: i.e. neighbouring chromosomes have similar fitnesses. In the case of the evolutionary strategy, the concept of a neighbourhood is present in the form of a mutation probability density function, with the average number of chromosomes mutated being related inversely to chromosome fitness. The smoothness property seems to hold for the problem studied here—in that the techniques that explore well-defined neighbourhoods performed better (the NAHC, tabu search and, to a lesser extent, the SAHC). However, considering that it had not been optimized in any way, the evolutionary strategy gave very encouraging results: out-performing the GA with more traditional parameter settings (GA2) in

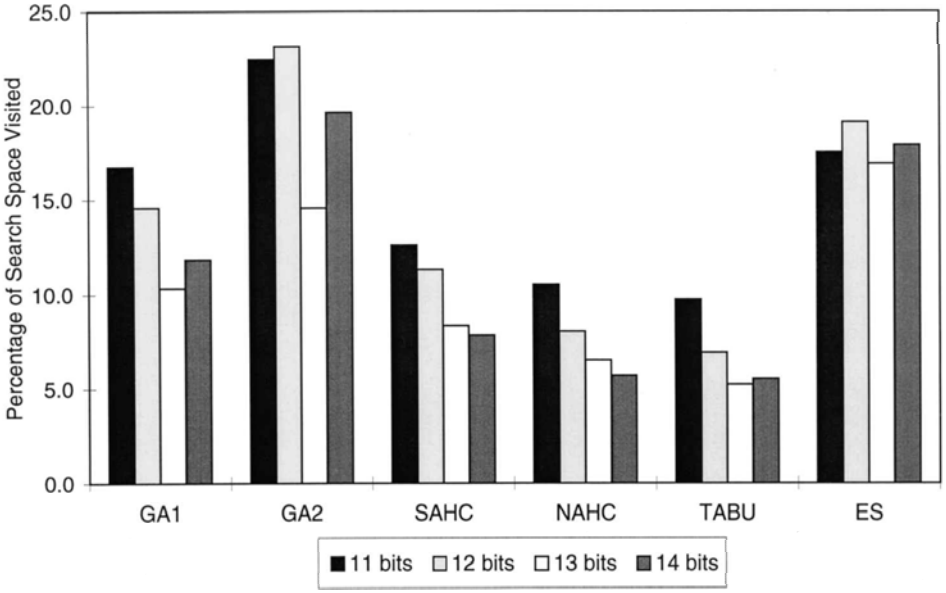

Figure 3. Mean percentage of search space visited to find the best fitness.

every case and giving results roughly equivalent to the partly optimized GA (GA1) in those experiments where the numbers of evaluations had been fixed.

In Fig. 3. it can be seen that for the SAHC, NAHC and tabu search algorithms, the mean percentage of search space visited to find the best, consistently decreased with an increasing number of variables. This is probably due to the fact that as the size of the search space increases there is a decreasing probability of the algorithms looking at chromosomes that were located at previous local optima. A study of how the numbers and location of optima vary with search space size in these problems might be revealing in this context. In contrast to this, the evolutionary algorithms (GA and ES) showed no clear trend toward either improvement or deterioration as the number of input variables increased.

As the number of chromosome evaluations decreased, the performance of the tabu search deteriorated faster than the other algorithms, particularly the SAHC and NAHC hill-climbers. Tabu search moves off local optima along the best path it can find, thus continuing to attempt to exploit previously found regularity in the search space. It seems that using this strategy to move between optima becomes less important than widely covering the search space when the number of evaluations is reduced.

The clear difference in performance of the NAHC and SAHC is consistent with the findings of other researchers that the choice of hill-climbing algorithm may be as significant as the choice between hill-climber and evolutionary algorithm. For example Forrest and Mitchell (1993) found the performance of their random mutation hill-climber (RMHC) vastly superior to both the NAHC and SAHC on a set of theoretical fitness functions (Royal Road) designed to study GA performance. Unfortunately, the RMHC is of no use in practical search spaces due to its inability to escape local optima. However, it is also worth noting that the performance of the RMHC was an order of magnitude better than a GA on the Royal Road benchmarks,

despite these having been specifically designed to be the simplest set of functions on which a GA should perform better than a hill-climber.

Although the performance of the GA was poor in comparison with the neighbourhood search methods (NAHC, SAHC and TS) tested here, previous results show that it is by no means a random search (Miller *et al.* 1994). An explanation of the success of the GA in general, and the relative performances of the GA with the two sets of parameters is required. The action of a GA has traditionally been explained by the building-block hypothesis (Holland 1968, 1975, Goldberg, 1989). This rests on the idea of a schema, a set of values for a group of bits, where these bits are not necessarily contiguous on the chromosome. The hypothesis states that GAs are successful when small highly-fit schemas combine to form even more highly fit, larger schemas. Selection causes fitter schemas to be present in greater numbers as the generations progress, and crossover combines these to form larger, super-fit schemas with an increasing probability. Mutation is regarded as a less important 'background' operator whose role is to aid in the formation of good schemas not present in the original population and to recover those fit schemas that are lost through statistical drift due to the stochastic nature of the selection process. Two commonly observed features of GAs are: (i) that they are less likely to get stuck in local optima than other methods, and (ii) when they have found a region containing good solutions they find it difficult to identify the best chromosome owing to the low probability of small moves. Given this, how then do GAs find optima at all? Reeves (1994) answers this question by considering GAs as a form of neighbourhood search. He pointed out that close to convergence (population homogeneity) the strings become more similar and so the average distance between parents decreases and thus the average move length decreases. Thus, when a population is close to convergence it acts more like a traditional hill-climber. This suggests an explanation for the relative success of the two sets of parameter values we used.

The more successful GA (GA1) had a far smaller population and higher mutation rate than GA2; this, combined with the very high selection pressure we used in our selection scheme, caused the smaller population GA1 to converge far more often than GA2, although both typically converged several times on each problem. When convergence has taken place, mutation provides the only mechanism for further exploration of the search space and it seems likely that this was why a higher mutation rate was found to give slightly improved results in our previous investigations. Convergence is almost universally regarded as being a bad thing by the GA community, who usually refer to it as premature convergence. Indeed there is much current research on GA mechanisms to avoid convergence (e.g. Miller 1994).

6. Conclusions

The search space of polarities of Reed–Muller canonical expansions was found, on average, to be quite tractable by neighbourhood search (NS) methods. It is possible that the NS methods used might be made even more efficient by refinement. In the case of the hill-climbers this would involve investigating heuristics to give a good starting polarity and to take account of the positions of previous optima when restarting the search after finding an optimum. In the case of tabu search one might try adding some of the poorer solutions in a neighbourhood to the tabu list as well as previous best solutions. The tabu list would then combine the roles of reducing the amount of time spent in poorer parts of the search space and avoiding direct backtracking. The greedy hill-climber (SAHC) was not as successful as the NAHC,

so it might be worth investigating the effect of making the tabu search less greedy by accepting the first solution in the neighbourhood with greater fitness than at present, rather than always examining all neighbouring solutions before making a move.

It is interesting that the ability of GAs to exploit regularities in the polarity search space may be due to the fact that they spent much of their time 'teetering' on the edge of convergence. Keeping a GA in this condition is against orthodox practice. According to Goldberg and Deb (1991): 'Holland's connection (1973, 1975) of the *k*-armed bandit problem to the conflict between exploration and exploitation in selection still remains the only sensible theoretical abstraction of the general question'. The present debate in theoretical evolutionary computing circles about how to balance exploratory (wide-ranging) and exploitative (local) searches often centres around the roles of the crossover and mutation operators (Spears 1993) and the parental selection method (Goldberg and Deb 1991, Hancock 1994). We hope that our results will inspire interest in the use of small populations as a means of tipping the balance towards local search, where this is found to be appropriate. We would also like to encourage the use of fitness look-up tables as a means of comparing the performance of algorithms on the basis of search space coverage, rather than simply seeing them as an implementational convenience.

Our performance results emphasize that the far simpler NS methods should always be considered as an alternative to genetic, or other evolutionary algorithms. However, the most suitable non-exhaustive search algorithm will always depend on the nature of the input function being optimized. A GA is more general-purpose than an NS method: because of crossover it does not rely on the mathematical structure associated with a neighbourhood function. The performance of GAs has been shown to be better than that of hill-climbers when noise is present in the fitness values (e.g. Davis 1991a) as would be typical if these were derived from a physical system. A theoretical discussion of functions found easy by GAs, but on which the steepest-ascent hill-climber fails, can be found in Wilson (1991).

The fact that hill-climbers were more successful than evolutionary methods further suggests very low epistasis in the input functions and a high degree of mathematical structure. This should give some encouragement to those researching deterministic methods for minimizing Reed–Muller canonical expansions of Boolean functions.

References

Almaini, A. E. A., 1989, *Electronic Logic Systems*, second edition (Englewood Cliffs, NJ: Prentice Hall).

Almaini, A. E. A., Thomson, P., and Hanson, D., 1991, Tabular techniques for Reed–Muller logic. *International Journal of Electronics*, **70**, 23–24.

Back, T., Hoffmeister, F., and Schwefel, H.-P., 1991, A survey of evolutionary strategies. *Proceedings of the Fourth International Conference on Genetic Algorithms* (San Mateo: Morgan Kaufmann), pp. 2–9.

Battiti, R., and Tecciolli, G., 1992, Parallel biased search for combinatorial optimisation: genetic algorithms and TABU. *Microprocessors and Microsystems*, **16**, 351–367.

Besslich, Ph. W., 1983, Efficient computer method for EX-OR logic design. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **130**, 203–206.

Davidor, Y., 1991, Epistasis variance: a viewpoint on GA hardness. *Foundations of Genetic Algorithms* (San Mateo: Morgan Kaufmann), pp. 23–35.

Davis, L., 1991a, Bit-climbing, representational bias and test suite design. *Proceedings of the Fourth International Conference on Genetic Algorithms* (San Mateo: Morgan Kaufmann), pp. 18–23; (editor), 1991b, *Handbook of Genetic Algorithms* (New York: Van Nostrand Reinhold).

FORREST, S., and MITCHELL, M., 1993, Relative building block fitness and the building-block hypothesis. *Foundations of Genetic Algorithms 2* (San Mateo: Morgan Kaufmann), pp. 109–126.

GLOVER, F., 1989, Tabu search—Part 1, *Operational Research Society of America Journal of Computing*, **1**(3), 190–206; 1990, Tabu search—Part 2, *Ibid.*, **2**(1), 4–32.

GOLDBERG, D. E., 1989, *Genetic Algorithms in Search, Optimisation and Machine Learning* (Reading, Mass: Addison-Wesley).

GOLDBERG, D. E., and DEB, K., 1991, A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms* (San Mateo: Morgan Kaufmann), pp. 69–93.

GREEN, D. H., 1986, *Modern Logic Design* (Wokingham, U.K.: Addison-Wesley); 1987, Reed–Muller expansions of incompletely specified functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **134**, 228–236.

HARKING, B., 1990, Efficient algorithm for canonical Reed–Muller expansion of Boolean functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **137**, 366–370.

HANCOCK, P. J. B., 1994, An empirical comparison of selection methods in evolutionary algorithms. *Lecture Notes in Computer Science 865: Evolutionary Computing* (Berlin: Springer-Verlag) pp. 80–94.

HOLLAND, J. H., 1968, Hierarchical Descriptions of Universal Spaces and Adaptive Systems. Technical Report ORA Projects 01252 and 08226, Department of Computer and Communication Sciences, University of Michigan, Ann Arbor, U.S.A.; 1973, Genetic algorithms and the optimal allocation of trials. *SIAM Journal of Computing, 2*, **2**, 88–105; 1975, *Adaptation in Natural and Artificial Systems* (Ann Arbor, Mich.: University of Michigan Press).

LUI, P. K., and MUZIO, J. C., 1991, Boolean matrix transforms for the minimisation of modulo-2 canonical expansions. *IEEE Transactions on Computers*, **41**, 342–347.

MCKENZIE, L., ALMAINI, A. E. A., MILLER, J. F., and THOMSON, P., 1993, Optimization of Reed–Muller logic functions. *International Journal of Electronics*, **75**, 451–466.

MICHAELOWICZ, Z., 1992, *Genetic Algorithms + Data Structures = Evolutionary Programs* (Berlin: Springer-Verlag).

MILLER, G. F., 1994, Exploiting mate choice in evolutionary computation: sexual selection as a process of search, optimisation and diversification. *Lecture Notes in Computer Science 865: Evolutionary Computing* (Berlin: Springer-Verlag), pp. 65–79.

MILLER, J. F., LUCHIAN, H., BRADBEER, P. V. G., and BARCLAY, P. J., 1994, Using a genetic algorithm for optimizing fixed polarity Reed–Muller expansions of Boolean functions. *International Journal of Electronics*, **76**, 601–609.

MILLER, J. F., and THOMSON, P., 1994, A highly efficient exhaustive search algorithm for optimising canonical Reed–Muller expansions of Boolean functions. *International Journal of Electronics*, **76**, 37–56.

MUKHOPADHYAY, A., and SCHMITZ, G., 1970, Minimisation of exclusive-Or and logical equivalence switching circuits. *IEEE Transactions on Computers*, **19**, 132–140.

REDDY, S. M., 1972, Easily testable realisations for logic functions. *IEEE Transactions on Computers*, **21**, 1183–1188.

RECHENBERG, I., 1973, *Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution* (Stuttgart, Germany: Fromann-Helzboog, Verlag).

REEVES, C. R. (Editor) 1993, *Modern Heuristic Methods for Combinatorial Problems* (Oxford, U.K.: Blackwell Scientific).

REEVES, C., 1994, Genetic algorithms and neighbourhood search. *Lecture Notes in Computer Science 865: Evolutionary Computing* (Berlin: Springer-Verlag) 115–130.

SARABI, A., and PERKOWSKI, M. A., 1992, Fast exact and quasi-minimal minimisation of highly testable fixed polarity AND/XOR canonical networks. *Proceedings of the 29th ACM/IEEE Design Automation Conference*, pp. 30–35.

SPEARS, W. M., 1993, Crossover or mutation? *Foundations of Genetic Algorithms 2* (San Mateo: Morgan Kaufmann), pp. 221–237.

TRAN, A., 1987, Graphical method for the conversion of minterms to Reed–Muller coefficients and the minimisation of EX–OR switching functions. *Proceedings of the Institution of Electrical Engineers*, Pt E, *Computers and Digital Techniques*, **134**, 93–99.

WILSON, S. W., 1991, GA-easy does not imply steepest-ascent optimisable. *Proceedings of the Fourth International Conference on Genetic Algorithms* (San Mateo: Morgan Kaufmann), pp. 85–89.